PATENT ABSTRACTS OF JAPAN

(11)Publication number : 2002-099418
(43)Date of publication of application : 05.04.2002

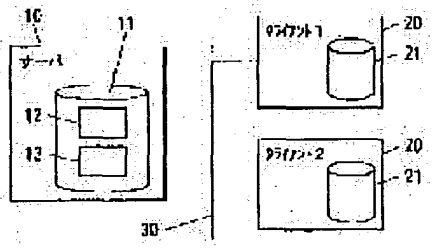
(51)Int.Cl. G06F 9/445
G06F 12/00

(21)Application number : 2000-290795
(22)Date of filing : 25.09.2000

(71)Applicant : SANYO ELECTRIC CO LTD
(72)Inventor : TANAKA YOSHIAKI

(54) VERSION UP METHOD

(57)Abstract:
PROBLEM TO BE SOLVED: To attain a prevention of version up in progress at a client side without completing version up to the latest version at a sitting in spite of existence of version up data of more than one generation in a server.
SOLUTION: In a client/server system, the version up data of software to be installed by a client is registered into the server going into two or more versions, the version up data is managed with connecting in order of version, a client performs version up successively for the version up data from the version of the software owned by the client up to a desired version of the generation in progress referring to the version up data registered into the server.



LEGAL STATUS

[Date of request for examination]

[Date of sending the examiner's decision of rejection]

[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]

[Date of final disposal for application]

[Patent number]

[Date of registration]

[Number of appeal against examiner's decision of rejection]

[Date of requesting appeal against examiner's decision of rejection]

[Date of extinction of right]

* NOTICES *

1. This document has been translated by computer.So the translation may not reflect the original precisely.
2. **** shows the word which can not be translated.
3.In the drawings, any words are not translated.

CLAIMS

[Claim(s)]
[Claim 1] It is the upgrade method which registers into a server the version updater of the software which a client introduces over two or more versions in a client/server system, connects the version updater in order of a version, manages it, and is characterized by a client upgrading one by one about the version updater after the version of self-client software to the version of a request of a generation the middle with reference to the version updater registered into the server.
[Claim 2] A client is the upgrade method according to claim 1 characterized by upgrading with reference to the version updater registered into the server to the version the chart example, among those the user specified the version updater after the version of self-client software to be.

[Translation done.]

DETAILED DESCRIPTION

[Detailed Description of the Invention]
[0001]
[The technical field to which invention belongs] In the LAN system of a client/server method, this invention manages the version updater of the software of a client unitary on a server, and relates to the method of upgrading client software.
[0002]
[Description of the Prior Art] In the client/server system, the version updater of two or more generations was managed by the server side, and, as for the applicant for this patent, the client proposed the method of upgrading to a latest version by Japanese Patent Application No. 10-19402 suitably. This registers into a server the table which connected the version updater of two or more generations in order of the version, and a client will carry it out sequentially from the version of lower order, if the data of a version higher than an own version are found with reference to the table. It enables it to upgrade by not leaking also about an intermediate version by this.
[0003]
[Problem(s) to be Solved by the Invention] Even when the version updater of two or more generations is in a server, in some cases, I want to stop by the intermediate version in a client side, rather than to upgrade at a stretch to the newest version. For example, it is a case so that the software may change the content and may be upgraded according to a statute. Although a new version comes out whenever a statute is revised, when the data which should be processed by the old statute remain in the client, it upgrades to the version corresponding to the old statute, and the upgrade after it is stopped. To be able to upgrade them to the version of each request individually, since situations differ is desired by the client of each [this] again.
[0004]
[Means for Solving the Problem] this invention registers into a server the version updater of the software which a client introduces over two or more versions in a client/server system, the version updater is connected in order of a version, it manages it, and with reference to the version updater registered into the server, a client is upgrading one by one about the version updater after the version of self-client software to the version of a request of a generation the middle, and solves the above-mentioned technical problem.
[0005]
[Example] Drawing 1 shows the outline composition of the client/server system which enforces the upgrade method concerning this invention. 10 -- a server machine -- it is -- the storage 11, such as a magnetic disk, -- having -- data and a program peculiar to a server there -- in addition, the upgrade data table 12 which memorized the version updater which a client carries out further, and the client managed table 13 which manages the version of the present of each client are memorized
[0006] 20 is a client machine and builds in the storage 21, such as a magnetic disk, like a server 10. The software of the program which operates on a client, data, etc. is memorized by this storage 21. And the version updater of a server 10 is incorporated through the LAN circuit 30 if needed, and upgrade of client software is performed. the client 20 of that one or more clients 20

can connect with the LAN circuit 30, and each -- it upgrades suitably independently, respectively
[0007] The example of composition of the upgrade data table 12 memorized by the server 10 is shown in drawing 4 . "Degree the address" which there is a link information in the head of a table and points out the address of the table which follows, and the "before address" which points out the address of the table to precede are memorized. Thereby, each version updater is connected in order of a version. And the substance of version updater is memorized behind a link information. However, only a top table is special and consists of only link informations. As shown in drawing, the following address of a top table is "B", and since the table which precedes the last address does not exist, on the other hand, "termination" code is memorized. On the 2nd table which follows, the following address is the address "C" of the 3rd table, and the last address is "A." And version updater, such as a program and data, is remembered to be version information, such as a version number, as substance of version updater. The 3rd table turns into a table of a tail in this example, and is this, i.e., the newest version updater. Therefore, the following address is "termination." The last address is the address "B" of the 2nd table to precede.
[0008] With reference to this upgrade data table 12 in the storage 11 of a server 10, by following the following address in a link information, a client 20 can check the version information on all the version updater stored in the server 10, and can download a desired version.
[0009] Moreover, the client managed table 13 is matched with a client number peculiar to two or more clients 20, and the version information on the client software introduced into each client 20 now is memorized. Client 20 refernce is possible also for the client managed table 13, and a client 20 updates its version information, when upgrading. [of the client managed table 13]
[0010] Next, operation of an example is explained. The case where a client 20 upgrades is explained with reference to drawing 2 . A client 20 investigates the version of all the version updater connected there with reference to the upgrade data table 12 of a server 10 (Step S1). Specifically, a head table is set as a current table and the following address judges whether it is termination. If it is not termination, the consecutive table will be set as the current table, the version information on a current table will be compared with the version of own client software, and it will judge whether upgrade is required. That is, it judges whether the version of a current table is newer than that [a client's] own, and if new, upgrade will need and it will add to a list. Then, a consecutive table is set as a current table, and upgrade completes a required list repeatedly until the following address becomes termination about this judgment.
[0011] And it is newer than the version of its software, namely, the version number of a high order is indicated by list (S2). The example of a display is shown in drawing 3 . Here, it is shown that there are a version "1.1", "1.2", and "1.5" as version updater which can be carried out. Moreover, it is displayed on right-hand side the thing corresponding to which statute each is. In addition, this information is memorized by the version information on the upgrade data table 12. Then, a user specifies how far the version of client software is raised, and directs implementation of upgrade (S3). For example, if you want to raise to a version "1.2", the inverse video of the 2nd version "1.2" will be carried out, and an operation button will be clicked. However, it is not necessary to carry out a low-ranking version "1.1" first also in [it] this case.
[0012] According to this, a client 20 upgrades sequentially from a low-ranking version to the appointed version (S4). In the example mentioned above, the version updater of a version "1.1" is incorporated and carried out first, and, next, the version updater of a version "1.2" is incorporated and carried out. And with reference to the client managed table 13 of a server 10, the version information corresponding to an own client number is updated to the upgraded version (S5).
[0013] Thus, with reference to the version of the version updater stored by connecting on the storage 11 of a server 10, whenever a thing newer than the own version of a client is found in a client, it repeats upgrade. A new thing is back connected sequentially from the old thing of a version, and version updater has the newest version updater in a tail. A client 20 upgrades in order using the version updater of all the found generations until it results in the version which the user specified after it started from the version updater of the oldest version in a server 10, it

ignored one by one as compared with the own version while they were old, and the thing newer than an own version appeared. consequently, the old generation's difference which is not contained in a new version when two or more generations are overdue and it upgrades -- it does not leak, and data can also be incorporated and upgraded

[0014] By the way, a server 10 can delete the version updater which became unnecessary as follows by referring to the client managed table 13. That is, the version information on the bottom grade of the upgrade data table 12 is compared with the version information on all the clients memorized by the client managed table 13, and any client 20 judges whether it is upgrade ending to the version. Here, the version of a client 20 is newer than it, or when the same, it judges with it being already updating ending to the version. And when finishing [updating], the version updater of the version is deleted from the upgrade data table 12.

[0015] By the way, upgrade processing in a client can completely be independently carried out also in the relation between clients also in a relation with a server. That is, storing processing of the new version in a server and the deletion of an old version, and upgrade processing of a client do not need to take a synchronization, and may be carried out to arbitration, respectively. Moreover, it is not necessary to take a synchronization also in both clients and to perform upgrade processing. Each client should just carry out arbitrarily.

[0016]

[Effect of the Invention] When the version updater of two or more generations is registered into the server, a client can be upgraded to a desired version according to each situation. Even if it is a case so that the software may change the content and may be upgraded according to a statute, when the data which should be processed by the old statute remain in the client by this, for example, it upgrades to the version corresponding to the old statute, and it becomes possible to stop the upgrade after it.

[Translation done.]

* NOTICES *

1.This document has been translated by computer. So the translation may not reflect the original precisely.
2.**** shows the word which can not be translated.
3.In the drawings, any words are not translated.

DETAILED DESCRIPTION

[Detailed Description of the Invention]
[0001]
[The technical field to which invention belongs] In the LAN system of a client/server method, this invention manages the version updater of the software of a client unitary on a server, and relates to the method of upgrading client software.
[0002]
[Description of the Prior Art] In the client/server system, the version updater of two or more generations was managed by the server side, and, as for the applicant for this patent, the client proposed the method of upgrading to a latest version by Japanese Patent Application No. 10-19402 suitably. This registers into a server the table which connected the version updater of two or more generations in order of the version, and a client will carry it out sequentially from the version of lower order, if the data of a version higher than an own version are found with reference to the table. It enables it to upgrade by not leaking also about an intermediate version by this.
[0003]
[Problem(s) to be Solved by the Invention] Even when the version updater of two or more generations is in a server, in some cases, I want to stop by the intermediate version in a client side, rather than to upgrade at a stretch to the newest version. For example, it is a case so that the software may change the content and may be upgraded according to a statute. Although a new version comes out whenever a statute is revised, when the data which should be processed by the old statute remain in the client, it upgrades to the version corresponding to the old statute, and the upgrade after it is stopped. To be able to upgrade them to the version of each request individually, since situations differ is desired by the client of each [this] again.
[0004]
[Means for Solving the Problem] this invention registers into a server the version updater of the software which a client introduces over two or more versions in a client/server system, the version updater is connected in order of a version, it manages it, and with reference to the version updater registered into the server, a client is upgrading one by one about the version updater after the version of self-client software to the version of a request of a generation the middle, and solves the above-mentioned technical problem.
[0005]
[Example] Drawing 1 shows the outline composition of the client/server system which enforces the upgrade method concerning this invention. 10 -- a server machine -- it is -- the storage 11, such as a magnetic disk, -- having -- data and a program peculiar to a server there -- in addition, the upgrade data table 12 which memorized the version updater which a client carries out further, and the client managed table 13 which manages the version of the present of each client are memorized
[0006] 20 is a client machine and builds in the storage 21, such as a magnetic disk, like a server 10. The software of the program which operates on a client, data, etc. is memorized by this storage 21. And the version updater of a server 10 is incorporated through the LAN circuit 30 if needed, and upgrade of client software is performed. the client 20 of that one or more clients 20

can connect with the LAN circuit 30, and each -- it upgrades suitably independently, respectively
[0007] The example of composition of the upgrade data table 12 memorized by the server 10 is shown in drawing 4. "Degree the address" which there is a link information in the head of a table and points out the address of the table which follows, and the "before address" which points out the address of the table to precede are memorized. Thereby, each version updater is connected in order of a version. And the substance of version updater is memorized behind a link information. However, only a top table is special and consists of only link informations. As shown in drawing, the following address of a top table is "B", and since the table which precedes the last address does not exist, on the other hand, "termination" code is memorized. On the 2nd table which follows, the following address is the address "C" of the 3rd table, and the last address is "A." And version updater, such as a program and data, is remembered to be version information, such as a version number, as substance of version updater. The 3rd table turns into a table of a tail in this example, and is this, i.e., the newest version updater. Therefore, the following address is "termination." The last address is the address "B" of the 2nd table to precede.
[0008] With reference to this upgrade data table 12 in the storage 11 of a server 10, by following the following address in a link information, a client 20 can check the version information on all the version updater stored in the server 10, and can download a desired version.
[0009] Moreover, the client managed table 13 is matched with a client number peculiar to two or more clients 20, and the version information on the client software introduced into each client 20 now is memorized. Client 20 refernce is possible also for the client managed table 13, and a client 20 updates its version information, when upgrading. [of the client managed table 13]
[0010] Next, operation of an example is explained. The case where a client 20 upgrades is explained with reference to drawing 2. A client 20 investigates the version of all the version updater connected there with reference to the upgrade data table 12 of a server 10 (Step S1). Specifically, a head table is set as a current table and the following address judges whether it is termination. If it is not termination, the consecutive table will be set as the current table, the version information on a current table will be compared with the version of own client software, and it will judge whether upgrade is required. That is, it judges whether the version of a current table is newer than that [a client's] own, and if new, upgrade will need and it will add to a list. Then, a consecutive table is set as a current table, and upgrade completes a required list repeatedly until the following address becomes termination about this judgment.
[0011] And it is newer than the version of its software, namely, the version number of a high order is indicated by list (S2). The example of a display is shown in drawing 3. Here, it is shown that there are a version "1.1", "1.2", and "1.5" as version updater which can be carried out. Moreover, it is displayed on right-hand side the thing corresponding to which statute each is. In addition, this information is memorized by the version information on the upgrade data table 12. Then, a user specifies how far the version of client software is raised, and directs implementation of upgrade (S3). For example, if you want to raise to a version "1.2", the inverse video of the 2nd version "1.2" will be carried out, and an operation button will be clicked. However, it is not necessary to carry out a low-ranking version "1.1" first also in [it] this case.
[0012] According to this, a client 20 upgrades sequentially from a low-ranking version to the appointed version (S4). In the example mentioned above, the version updater of a version "1.1" is incorporated and carried out first, and, next, the version updater of a version "1.2" is incorporated and carried out. And with reference to the client managed table 13 of a server 10, the version information corresponding to an own client number is updated to the upgraded version (S5).
[0013] Thus, with reference to the version of the version updater stored by connecting on the storage 11 of a server 10, whenever a thing newer than the own version of a client is found in a client, it repeats upgrade. A new thing is back connected sequentially from the old thing of a version, and version updater has the newest version updater in a tail. A client 20 upgrades in order using the version updater of all the found generations until it results in the version which the user specified after it started from the version updater of the oldest version in a server 10, it

ignored one by one as compared with the own version while they were old, and the thing newer than an own version appeared. consequently, the old generation's difference which is not contained in a new version when two or more generations are overdue and it upgrades -- it does not leak, and data can also be incorporated and upgraded

[0014] By the way, a server 10 can delete the version updater which became unnecessary as follows by referring to the client managed table 13. That is, the version information on the bottom grade of the upgrade data table 12 is compared with the version information on all the clients memorized by the client managed table 13, and any client 20 judges whether it is upgrade ending to the version. Here, the version of a client 20 is newer than it, or when the same, it judges with it being already updating ending to the version. And when finishing [updating], the version updater of the version is deleted from the upgrade data table 12.

[0015] By the way, upgrade processing in a client can completely be independently carried out also in the relation between clients also in a relation with a server. That is, storing processing of the new version in a server and the deletion of an old version, and upgrade processing of a client do not need to take a synchronization, and may be carried out to arbitration, respectively. Moreover, it is not necessary to take a synchronization also in both clients and to perform upgrade processing. Each client should just carry out arbitrarily.

[0016]

[Effect of the Invention] When the version updater of two or more generations is registered into the server, a client can be upgraded to a desired version according to each situation. Even if it is a case so that the software may change the content and may be upgraded according to a statute, when the data which should be processed by the old statute remain in the client by this, for example, it upgrades to the version corresponding to the old statute, and it becomes possible to stop the upgrade after it.

[Translation done.]

* NOTICES *

1.This document has been translated by computer. So the translation may not reflect the original precisely.
2.**** shows the word which can not be translated.
3.In the drawings, any words are not translated.

DESCRIPTION OF DRAWINGS

[Brief Description of the Drawings]
[Drawing 1] It is the block diagram showing the composition of the client/server system of an example.
[Drawing 2] It is drawing showing the upgrade procedure of a client.
[Drawing 3] It is the example of a list display of the version which can be upgraded.
[Drawing 4] It is drawing showing the upgrade data table of an example.
[Description of Notations]
10 Server 11 Storage 12 Upgrade Data Table
13 Client Managed Table 20 Client 21 Storage 30 LAN Circuit

[Translation done.]

(19)日本国特許庁（ＪＰ）　(12)公開特許公報（Ａ）　(11)特許出願公開番号
特開2002－99418
(P2002－99418A)
(43)公開日　平成14年４月５日(2002.4.5)

(51)Int.Cl.[7]	識別記号	ＦＩ		テーマコード(参考)
Ｇ０６Ｆ　9/445		Ｇ０６Ｆ　12/00	５１７	５Ｂ０７６
12/00	５１７	9/06	６１０Ｑ	５Ｂ０８２

審査請求　未請求　請求項の数２　ＯＬ　（全　４　頁）

(21)出願番号　特願2000－290795(P2000－290795)

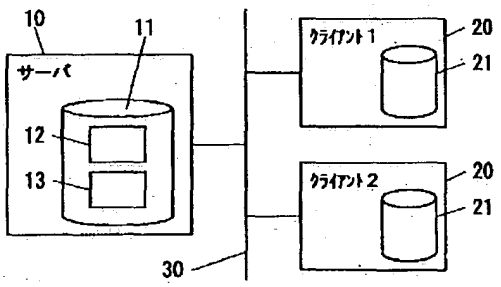
(22)出願日　平成12年９月25日(2000.9.25)

(71)出願人　000001889
三洋電機株式会社
大阪府守口市京阪本通２丁目５番５号
(72)発明者　田中　義昭
大阪府守口市京阪本通２丁目５番５号　三洋電機株式会社内
(74)代理人　100111383
弁理士　芝野　正雅

Ｆターム(参考)　5B076 AC03
5B082 GA05 GA14

(54)【発明の名称】　バージョンアップ方法

(57)【要約】
【課題】　サーバに複数世代のバージョンアップデータがあるときでも、クライアント側では、一気に最新のバージョンまでバージョンアップするのではなく、途中のバージョンで止めておきたいこともある。
【解決手段】　クライアント／サーバシステムにおいて、クライアントが導入するソフトウェアのバージョンアップデータを複数バージョンにわたってサーバに登録し、そのバージョンアップデータをバージョンの順に連結して管理し、クライアントはサーバに登録されたバージョンアップデータを参照して、途中世代の所望のバージョンまで、自クライアントソフトウェアのバージョン以降のバージョンアップデータについて順次バージョンアップを実施するようにした。



【特許請求の範囲】
【請求項１】　クライアント／サーバシステムにおいて、クライアントが導入するソフトウェアのバージョンアップデータを複数バージョンにわたってサーバに登録し、そのバージョンアップデータをバージョンの順に連結して管理し、クライアントはサーバに登録されたバージョンアップデータを参照して、途中世代の所望のバージョンまで、自クライアントソフトウェアのバージョン以降のバージョンアップデータについて順次バージョンアップを実施することを特徴とするバージョンアップ方法。
【請求項２】　クライアントはサーバに登録されたバージョンアップデータを参照し、自クライアントソフトウェアのバージョン以降のバージョンアップデータを一覧表示し、そのうちユーザが指定したバージョンまでバージョンアップを実施することを特徴とする請求項１に記載のバージョンアップ方法。
【発明の詳細な説明】
【０００１】
【発明の属する技術分野】本発明は、クライアント／サーバ方式のＬＡＮシステムにおいて、クライアントのソフトウェアのバージョンアップデータをサーバ上で一元的に管理し、クライアントソフトウェアのバージョンアップを行う方法に関する。
【０００２】
【従来の技術】クライアント／サーバシステムにおいて、複数世代のバージョンアップデータをサーバ側で管理し、クライアントが適宜、最新バージョンにバージョンアップできる方法を、本願出願人は特願平１０－１９４０２にて提案した。これは、複数世代のバージョンアップデータを、バージョンの順に連結したテーブルをサーバに登録し、クライアントはそのテーブルを参照して、自身のバージョンより高いバージョンのデータを見つけると、それを低位のバージョンから順に実施するものである。これにより、途中のバージョンについても漏れなくバージョンアップを実施できるようにしたものである。
【０００３】
【発明が解決しようとする課題】サーバに複数世代のバージョンアップデータがあるときでも、クライアント側では、一気に最新のバージョンまでバージョンアップするのではなく、途中のバージョンで止めておきたいこともある。例えば、そのソフトウェアが法令に従って内容を変更し、バージョンアップされるような場合である。法令が改正されるごとに新しいバージョンが出るが、旧法令で処理すべきデータがクライアントに残っているときは、その旧法令に対応するバージョンまでバージョンアップを実施し、それ以降のバージョンアップは休止しておく。これはまた、個々のクライアントによって事情は異なるので、それぞれが所望のバージョンに個別にバージョンアップできることが望まれる。
【０００４】
【課題を解決するための手段】本発明は、クライアント／サーバシステムにおいて、クライアントが導入するソフトウェアのバージョンアップデータを複数バージョンにわたってサーバに登録し、そのバージョンアップデータをバージョンの順に連結して管理し、クライアントはサーバに登録されたバージョンアップデータを参照して、途中世代の所望のバージョンまで、自クライアントソフトウェアのバージョン以降のバージョンアップデータについて順次バージョンアップを実施することで、上記課題を解決するものである。
【０００５】
【実施例】図１は、本発明に係るバージョンアップ方法を実施するクライアント／サーバシステムの概略構成を示している。１０はサーバマシンであり、磁気ディスク等の記憶装置１１を持ち、そこには、サーバ固有のデータやプログラムに加えて、さらにクライアントが実施するバージョンアップデータを記憶したバージョンアップデータテーブル１２と、クライアント各々の現在のバージョンを管理するクライアント管理テーブル１３とを記憶している。
【０００６】２０はクライアントマシンであり、サーバ１０と同様に磁気ディスク等の記憶装置２１を内蔵している。この記憶装置２１にはクライアント上で動作するプログラムやデータ等のソフトウェアが記憶されている。そして、必要に応じてＬＡＮ回線３０を介してサーバ１０のバージョンアップデータを取込み、クライアントソフトウェアのバージョンアップが行われる。１つまたは複数のクライアント２０がＬＡＮ回線３０に接続可能であり、個々のクライアント２０それぞれが、独立にバージョンアップを適宜実施するものである。
【０００７】サーバ１０に記憶されたバージョンアップデータテーブル１２の構成例を図４に示す。テーブルの先頭にはリンク情報があり、後続するテーブルのアドレスを指す「次アドレス」と、先行するテーブルのアドレスを指す「前アドレス」が記憶される。これにより、各バージョンアップデータがバージョンの順に連結される。そして、リンク情報の後ろにバージョンアップデータの実体が記憶される。但し、先頭のテーブルだけは特別で、リンク情報のみで構成されている。図に示すように、先頭のテーブルの次アドレスは「Ｂ」であり、一方前アドレスは、先行するテーブルが存在しないので「終端」コードが記憶されている。後続する２番目のテーブルでは、次アドレスが３番目のテーブルのアドレス「Ｃ」であり、前アドレスは「Ａ」である。そして、バージョンアップデータの実体として、バージョン番号等のバージョン情報と、プログラムやデータ等のバージョンアップデータが記憶されている。３番目のテーブルはこの例では末尾のテーブルになり、これは即ち最新のバ

ージョンアップデータである。従って、次アドレスは「終端」である。前アドレスは先行する２番目のテーブルのアドレス「B」である。

【０００８】クライアント２０は、サーバ１０の記憶装置１１にあるこのバージョンアップデータテーブル１２を参照し、リンク情報の中の次アドレスを追っていくことによって、サーバ１０に格納されているすべてのバージョンアップデータのバージョン情報を確認することができ、所望のバージョンをダウンロード可能である。

【０００９】また、クライアント管理テーブル１３は、複数のクライアント２０に固有のクライアント番号に対応付けて、現在それぞれのクライアント２０に導入されているクライアントソフトウェアのバージョン情報を記憶する。クライアント管理テーブル１３もクライアント２０から参照可能であり、クライアント２０は、バージョンアップを実施した時、クライアント管理テーブル１３の自分のバージョン情報を更新するものである。

【００１０】次に、実施例の動作を説明する。クライアント２０がバージョンアップを実施する場合を、図２を参照して説明する。クライアント２０は、サーバ１０のバージョンアップデータテーブル１２を参照し、そこに連結されているすべてのバージョンアップデータのバージョンを調査する（ステップＳ１）。具体的には、先頭テーブルをカレントテーブルに設定し、その次アドレスが終端かどうか判定する。終端でなければ、後続のテーブルをカレントテーブルに設定しておいて、カレントテーブルのバージョン情報と自身のクライアントソフトウェアのバージョンとを比較し、バージョンアップが必要か否かを判定する。即ち、カレントテーブルのバージョンがクライアント自身のそれより新しいか否かを判定して、新しければバージョンアップが必要として、リストに加えるのである。その後、後続のテーブルをカレントテーブルに設定し、この判定を次アドレスが終端になるまで繰り返して、バージョンアップが必要なリストを完成する。

【００１１】そして、自分のソフトウェアのバージョンより新しい、即ち上位のバージョン番号を一覧表示する（Ｓ２）。その表示例を図３に示す。ここでは、実施可能なバージョンアップデータとしてバージョン「１．１」と「１．２」、「１．５」があることが示されている。また、それぞれがどの法令に対応するものかが、右側に表示されている。尚、この情報は、バージョンアップデータテーブル１２のバージョン情報に記憶されていたものである。そこで、ユーザは、クライアントソフトウェアのバージョンをどこまで上げるかを指定して、バージョンアップの実施を指示する（Ｓ３）。例えば、バージョン「１．２」まで上げたいのであれば、２番目のバージョン「１．２」を反転表示させておいて、実施ボタンをクリックする。但し、この場合にも、それより下位のバージョン「１．１」を先に実施しておく必要はない。

【００１２】これに応じて、クライアント２０は、指定のバージョンまで、下位のバージョンから順にバージョンアップを実施する（Ｓ４）。上述した例では、先ずバージョン「１．１」のバージョンアップデータを取り込んで実施し、次にバージョン「１．２」のバージョンアップデータを取り込んで実施する。そして、サーバ１０のクライアント管理テーブル１３を参照し、自身のクライアント番号に対応するバージョン情報を、バージョンアップしたバージョンに更新するのである（Ｓ５）。

【００１３】このようにクライアントは、サーバ１０の記憶装置１１上に連結して格納されているバージョンアップデータのバージョンを参照して、クライアント自身のバージョンより新しいものが見つかる度にバージョンアップを繰り返すのである。バージョンアップデータはバージョンの古いものから順に新しいものが後ろに連結され、末尾に最新のバージョンアップデータがある。クライアント２０は、サーバ１０にある最も古いバージョンのバージョンアップデータから開始して、１つ１つ自身のバージョンと比較してそれらが古い間は無視し、自身のバージョンより新しいものが現れてからは、ユーザが指定したバージョンに到るまで、見つかったすべての世代のバージョンアップデータを使って、順にバージョンアップを実施するのである。その結果、２世代以上遅れてバージョンアップする場合にも、新バージョンに含まれない旧世代の差分データをも漏れなく取込んでバージョンアップできるものである。

【００１４】ところで、クライアント管理テーブル１３を参照することで、サーバ１０は次のようにして不要になったバージョンアップデータを削除することができる。即ち、バージョンアップデータテーブル１２の最も下位のバージョン情報と、クライアント管理テーブル１３に記憶されている全クライアントのバージョン情報とを比較し、いずれのクライアント２０もそのバージョンにバージョンアップ済みであるか判定する。ここでは、クライアント２０のバージョンがそれより新しいか同じである場合は、そのバージョンに既に更新済みであると判定する。そして、更新済みの場合には、バージョンアップデータテーブル１２からそのバージョンのバージョンアップデータを削除するのである。

【００１５】ところで、クライアントにおけるバージョンアップ処理は、サーバとの関係においても、またクライアント相互の関係においても全く独立に実施できるものである。即ち、サーバにおける新バージョンの格納処理、及び古いバージョンの削除処理と、クライアントのバージョンアップ処理とは同期をとる必要はなく、それぞれ任意に実施してよい。また、クライアント相互においても同期を取ってバージョンアップ処理を行う必要はない。個々のクライアントが任意に実施すればよいものである。

【００１６】

【発明の効果】複数世代のバージョンアップデータがサーバに登録してある場合、クライアントはそれぞれの事情に応じて、所望のバージョンまでバージョンアップを実施できるようになる。これにより、例えば、そのソフトウェアが法令に従って内容を変更し、バージョンアップされるような場合であっても、旧法令で処理すべきデータがクライアントに残っているときには、その旧法令に対応するバージョンまでバージョンアップを実施し、それ以降のバージョンアップは休止しておくことが可能になる。

【図面の簡単な説明】

【図１】実施例のクライアント／サーバシステムの構成を示すブロック図である。

【図２】クライアントのバージョンアップ処理手順を示す図である。

【図３】バージョンアップ可能なバージョンの一覧表示の例である。

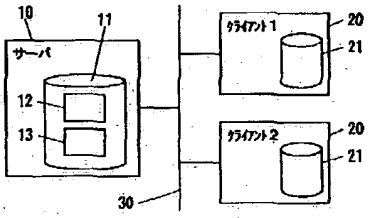
【図４】実施例のバージョンアップデータテーブルを示す図である。

【符号の説明】

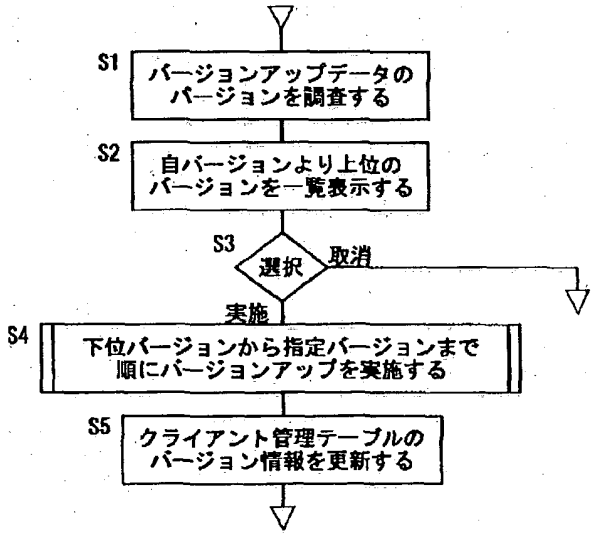
１０　サーバ　　１１　記憶装置　　１２　バージョンアップデータテーブル

１３　クライアント管理テーブル　　２０　クライアント　　２１　記憶装置　　３０　ＬＡＮ回線

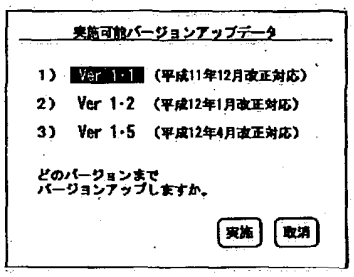
【図１】



【図２】



【図３】



【図４】